Towards Method Component Contextualization

Elena Kornyshova, Rebecca Deneckere, Bruno Claudepierre

Towards Method Component Contextualization

Elena Kornyshova, Rébecca Deneckère, Bruno Claudepierre
Centre de Recherche en Informatique,
Université Paris I – Panthéon Sorbonne, Paris, France

ABSTRACT
Method Engineering (ME) is a discipline which aims to bring effective solutions to the construction, improvement and modification of the methods used to develop Information Systems (IS). Situational Method Engineering (SME) promotes the idea of retrieving, adapting and tailoring components, rather than complete methodologies, to the specific context. Existing SME approaches use the notion of context for characterizing situations of IS development projects and for guiding the method components selection from a repository. However, in the reviewed literature, there is no proposed approach to specify the specific context of method components. This paper provides a detailed vision of context and a process for contextualizing methods in the IS domain. Our proposal is illustrated with three case studies: scenario conceptualization, project portfolio management and decision-making.

Keywords: Method engineering; Method component; Contextualization.

1. INTRODUCTION
An IS development methodology (ISDM) is a set of ideas, approaches, techniques and tools which system analysts use to help them transforming organizational needs into an appropriate Information System. The application areas of these methodologies are various. Because of this diversity, it is now apparent that a universal method that could be applied to deal with any IS development project does not exist. Method engineering (ME) represents the effort to improve the usefulness of ISDM by creating an adaptation framework whereby methods are created to match specific organizational situations. ME aims to find solutions to the construction, improvement and modification of the methods used to develop information systems. One of the ME fundamentals for optimizing, reusing, and ensuring flexibility and adaptability of these methods is their decomposition into modular parts (Harmsen, Brinkkemper & Han Oei 1994) (Rolland 2005). This purpose is the object of Situational Method Engineering (SME) which promotes the idea of retrieving, adapting and tailoring components, rather than complete methodologies, to the specific context.

Existing SME approaches consider the notion of context in order to guide the selection of a method component from a repository according to a given situation. They also deal with different kinds of context factors characterizing situations of IS development projects and offer various methodologies for using context. For instance, the method component context is studied in different approaches and is represented as: reuse frame (Mirbel 2008); interface (Ralyté & Rolland 2001b); method service context (Guzélian & Cauvet 2007); contingency factors (Van Slooten & Hodes 1996), (Harmsen 1997); development situation (Karlsson & Agerfalk 2004).

These approaches foresee different context elements which are the characteristics of method components.

However, the reviewed literature shows that, firstly, there is no approach considering all of the possible characteristics and, secondly, these approaches do not suggest a methodology allowing to define a set of concrete context characteristics for a given method.

In our view, the context is a set of characteristics which describes situations of a method application. The context is defined for an IS development method and its components. Each method component is then described by concrete values of these characteristics. In this paper, we focus on the contextualization of method components. Our goal is to propose (i) a generic model of context based on the state-of-the-art and (ii) an IS development methods contextualization process. We introduce the frame of contextualization, we present the context model, the context typology and the process to construct the context characteristics set for a given method. We illustrate our proposal with three case studies: scenario conceptualization, project portfolio management and decision-making.

All processes in this work are formalized with the MAP model which is commonly used in the ME field (Rolland, Prakah & Benjamen 1999). In our proposal, this formalism is used to represent the contextualization process in an intentional way. In the case studies, it is used to represent the organization of the method components (the links between them).

The paper is organized as follows. The notion of method component is described in the second section. Third section surveys a state-of-the-art on the notion of context. The fourth section proposes a context model and a process for the contextualization of method components. We illustrate our proposal with examples in the fifth section. Related works are given in the sixth section. A conclusion and future works are given in the last section.

2. CONTEXT AND ITS APPLICATION IN METHOD ENGINEERING

2.1. Cross domains application of Context-awareness

(Bouquet, Ghidini, Giunchiglia & Blanzieri 2003) states that the study of context was started in the 70's. Since then, many different domains in relation with information systems use the notion of context and give various interpretations of it. For instance, (Dey, Abowd & Salber 2001) defines the notion of context by the information that could be used for characterizing the situation of an entity (person, object or computer), and, more generally, by any element that can influence the IS behavior. (Rey & Coutaz 2002) foresees the context from four points of view:

- The context must be defined in terms of an object. It means that "there is no context without context".
- The capture of context is not the goal in itself but the captured data must serve a purpose.
- The context is an information space shared by multiple actors (users and systems).
- The context is infinite and varies with the passing of time.

Context models are multidisciplinary and have been proposed in several areas (Bradley & Dunlop 2005). The linguistic research is concerned with analyzing the usage context of signs (or words) within a language. Bunt (Bunt 1997) defines five types of context for communication aspects which are respectively:

- *Linguistic:* refers to linguistic material;
- *Semantic:* refers to domain description including objects and properties;

- *Physical:* refers to the environment description in which action or interaction occurs;
- *Social:* refers to the interactive situation which occurs between actors;
- *Cognitive:* refers to the participants' intentions, their evolution relating to perception, production, evaluation and execution.

Context is also formalized using mathematical models. For instance, (Coutaz & Rey 2002) proposes a cumulative model where the context (*Ctx*) is a timely aggregation of situations. A situation is a state descriptor for a user (*U*) performing a task (*T*) at a specific time (*t*). The model is depicted by the following formula:

$$Ctx(U,T,t) = \bigcup_{n=1}^{m} (Situation(U,T,t_n))$$

Related to the Information technologies field, the context is represented as a model or an ontology. For instance, (Gu, Wang, Pung & Zhang, 2004) suggests a more detailed vision of context. It describes a formal context model based on ontology for intelligent environments. This context ontology defines a vocabulary for representing knowledge about context in this field. It includes two levels: upper ontology (capturing general context knowledge) and domain-specific ontologies (detailing basic concepts in application to a given domain). (Gu, Wang, Pung & Zhang, 2004) also specifies a way for modeling context classification, dependency between context elements, and quality of context.

In the field of Knowledge Representation and Reasoning (KRR), which is an area of Artificial Intelligence, two types of the context theory have been proposed: (i) *divide-and-conquer*, which sees context as a way of partitioning a global model of the world into smaller and simpler pieces and (ii) *compose-and-conquer*, which sees context as a local theory of the world in a network of relations with other local theories (Bouquet, Ghidini, Giunchiglia & Blanzieri 2003).

Another term, closely related to the context one, is *context-awareness*. Context awareness is a term originating from pervasive computing, or ubiquitous computing (Schilit, Adams & Want 1994). These systems deal with linking changes in the environment with computer systems, which are otherwise static. Although it is a computer science term, it has also been applied to business theory in relation to business process management issues (Rosemann & Recker 2006).

There are numerous context-awareness applications when human interactions occur. More related to our study, context models are also proposed for business process reengineering (Bessai, Claudepierre, Saidani & Nurcan 2008), computer science (Bradley & Dunlop 2005), service selection (Kirsch Pinheiro, Vanrompay & Berbers 2008) and decision-making within a military situation (Rosen, Fiore, Salas, Letsky & Warner 2008), (Drury & Scott 2008). In latter cases, the context model is seen as a way to analyze a given *situation* to guide the way of processing. Thus, context models are mainly used to solve the problem of lacking flexibility and adaptability within processes.

2.2. Method Engineering and Method Components

Method Engineering is a discipline which aims to bring effective solutions to the construction, improvement and modification of the methods used to develop information and software systems. Several authors tried to design methods that would be as effective and as adapted as possible to the development needs of information systems (Firesmith & Henderson-Sellers 2001) (Rolland & Cauvet 1992). This goal was not always reached, especially because the methods were not always well adapted to projects specificities. The situational methods were designed to correct this weakness. The situational approach finds its justification in the practical field

analysis which shows that a method is never followed literally (Ralyte 2001) (Mirbel & de Rivieres 2002). Situational Method Engineering promotes the idea of using components, instead of complete methodologies, to specific situations (Ralyté & Rolland 2001a). In order to succeed in creating good methodologies that best suit given situations, components (building blocks of methodologies) representation and cataloguing are very important activities. In particular, the components have to be represented in a uniform way that includes all the necessary information that may influence their retrieval and assembling.

The notion of method component is central of SME as it promotes the idea of retrieving, adapting and tailoring modular parts, rather than complete methodologies, to specific situations. There are various representations of modular parts: fragments (Brinkkemper 1996), chunks (Rolland, Plihon & Ralyté 1998), components (Wistrand & Karlsson 2004), OPF fragments (Henderson-Sellers 2002) and method services (Deneckère, Iacovelli, Kornyshova & Souveyet 2008) (Guzélian & Cauvet 2007) (Iacovelli, Souveyet & Rolland 2008).

Method fragment approach (Brinkkemper 1996). Fragments are standardized building blocks based on a coherent part of method. A fragment is either a Product or a Process fragment and is stored on a method base from which they can be retrieved to construct a new method following assembly rules (Bunt 1997). The method component definition consists in encouraging a global analysis of the project while basing itself on contingency criteria. Projects and situations are characterized by means of factors associated with the methods.

Method chunk approach (Ralyté, Deneckère & Rolland 2003). A chunk is described as a way to capture more of the situational aspects in ME and to appropriately support the retrieval process. A chunk based method aims at associating the reusable components to their description in order to facilitate component research and extraction according to the user's needs. The chunk approach expresses projects requirements (the context) as a requirements map, which is used to test the similarity between requirements and existing components.

Method component (Wistrand & Karlsson 2004). Components allow viewing methods as constituted by exchangeable and reusable components. Each component consists of descriptions for process (rules and recommendations), notations (semantic, syntactic and symbolic rules for documentation), and concepts. This approach introduces the notion of method rationale which is the systematic treatment of the arguments and reasons behind a particular method. In the same way, the component description contains its rationale. Its matching with the context is performed by goal analysis.

OPF fragment (Henderson-Sellers 2002). In the OPEN Process Framework (*OPF*), the fragment is generated from an element in a prescribed underpinning meta-model. This meta-model has been upgraded with the availability of the international standard ISO/IEC 24744.

Method service (Guzélian & Cauvet 2007). This approach offers a repository with a large variety of method fragments, called method services, together with a service composition process. During composition, the process guides developer’s choices; it selects method services and delivers a method fragment that achieves developer’s requirements. The SO2M meta-model is based on three main principles: service orientation, task ontology for reuse of knowledge on development problems and dynamic construction of method services for generating tailored methods. The method service approach uses an identification part that defines the purpose of the service. The component retrieval is thus done by using goal, actor, process, and product ontologies.

(Deneckère, Iacovelli, Kornyshova & Souveyet 2008) structures the process of SME according to three steps of manipulating method components:

(a) the decomposition of methods into components which are stored in a method repository,
(b) the retrieval of components that better match the project specificities and
(c) the construction of a new method with these selected components.

According to these steps, different method components could be compared according to the four following criteria: decomposition principle, retrieval/selection principle, matching with situation, and construction technique (See Table 1).

First, the methods are decomposed into methods components which are stored in method base (or repository). Thus, we define the criterion "*decomposition principle*" which deals with different ways to decompose methods into components. This principle predefines the components' description used for their identification during project fulfilment.

Once the methods are decomposed and stored in the base, they could be used in the projects. On the first step, the engineer must find in the method base the components that better match the project specificities. On this basis, we identify two criteria: retrieval/selection principle and matching with situation. The *retrieval/selection principle* defines steps to carry out for identifying an appropriate component. In ME, all approaches are situational, which means they take into account the specific project situation by different manners. This aspect is considered within the *matching with situation* attribute.

The next step is to build a new method from the selected components. Based on (Nehan & Deneckère 2007), we distinguish the following main manners to use components for *constructing a new method* according to project specificities: assembly, extension, and reduction. By assembly, separate fragments are grouped with regard to the studied specific project to form a unique method (Ralyté, Deneckere & Rolland 2003). By applying extension, a basic method is transformed into a new one by addition of new components (Ralyté, Deneckere & Rolland 2003). By reduction, some components are removed from the basic method in order to transform it to match the engineer's needs (Wistrand & Karlsson 2004).

Table 1. Method Components Comparison.

Criteria	Fragment	Chunk	Component	OPF Fragment	Method Service
Decomposition principle		by intentions	by goal	inheritance, instantiation	Not specified
Retrieval/selection principle	Request	similarity measure	request by goal	request by goal	semantic similarity
Matching with situation	project characterisation	requirements map		by goal and actor	by goal, actor, process, and product ontologies
Construction technique	assembly	assembly, extension	assembly, extension, reduction	agile	assembly without overlapping

Decomposition Principle. The decomposition principle is quite different following the component type. Method fragment uses a tree decomposition to link all coherent method parts. Chunks are obtained by intentional decomposition of methods (Ralyté, Deneckere & Rolland 2003). The OPF fragment is a *clabject*, which is a result of both instantiation and inheritance (Gonzales-Perez 2007). Components are decomposed by goals (Wistrand & Karlsson 2004). The method service approach does not specify this attribute value.

Retrieval/Selection Principle. The retrieval and selection of a method fragment are made by different types of queries. Chunks are selected with the application of similarity measures of their descriptors and interfaces. This helps to evaluate the degree of matching between them and the requirements (Ralyté, Deneckere & Rolland 2003). On the same way, the method service selection is made by a comparison of the requirements (expressed by intentions) with the service intentional descriptors by ontologies, which allow comparing the semantic similarity (Guzélian & Cauvet 2007). Differently, OPF fragments, stored on a 'work product tool', are selected with queries on their endeavour (Gonzales-Perez 2007). Method fragments are selected by application of request on the goal (Harmsen, Brinkkemper & Oei 1994).

Matching with situation. Approaches don't match the situation with the same techniques. The method fragment definition consists in encouraging a global analysis of the project while basing itself on contingency criteria. Projects and situations are characterized by means of factors associated with the methods. The chunk approach includes projects requirements expressed as a *requirements map* (Ralyté, Deneckere & Rolland 2003), which is used to test the similarity between requirements and existing fragments. In component containing its "rational", the matching is performed by goal analysis (Wistrand & Karlsson 2004). The Method service approach uses an identification part that defines the purpose of the service. The matching is thus done by using goal, actor, process, and product ontologies (Guzélian & Cauvet 2007).

Construction technique. The method fragments are assembled for creating a new method. The chunk approach uses assembly (allowing overlapping between different chunks) and extension. In addition to the assembling and extending, the component approach suggests method reduction. The method service construction is based on a composition process that supports the aggregation of services in sequence or in parallel (Guzélian & Cauvet 2007). In the OPF approach, a new method is constructed by dynamic instantiation of fragments during the project. Hence, the OPF approach suggests an agile construction of methods.

A more detailed comparison of these different kinds of modular parts may be found in (Deneckère, Iacovelli, Kornyshova & Souveyet 2008).

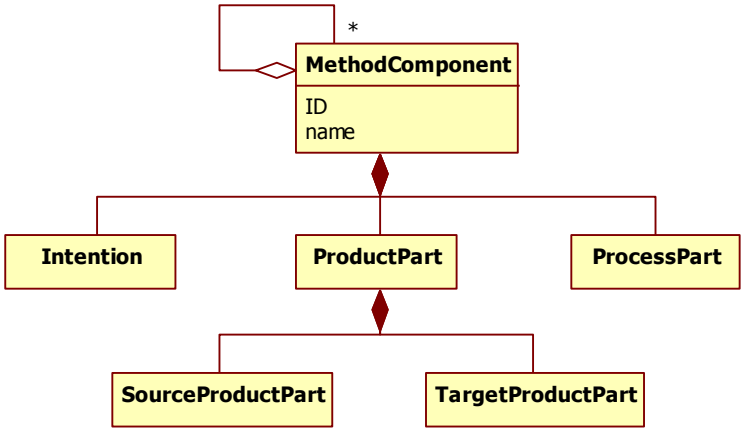

Figure 1. Method Component Meta-model.

Our view of a component has been described in (Deneckère, Iacovelli, Kornyshova & Souveyet 2008). We based our method component on the method chunk for its intrinsic intentionality as we decompose the methods into method components according to an intentional

principle. This part will then be used for retrieving and selecting method components from the method base. We suggest modelling method components as shown at Fig. 1.

Method components are expressed at different granularity, at various levels of abstraction. For instance, a component may be an entire method that can be decomposed into other less complex components (which, in turn, may also be decomposed into other more simple components, and so on). They are a representation of the components composition.

The *intention* describes the general purpose of the component. The *product part* corresponds to the description of the component input and output product models. The *source product part* defines the product required for applying the component. The *target product part* defines the result, which must be obtained by the component application. The *process part* contains guidelines which explain how to apply the component in order to obtain the product part.

For instance, a method component *Weighting* is given at Fig. 2. for illustrating this model. This component is part of decision-making methods. It allows defining weights to criteria in a given decision-making situation. The Weighting component is described by its ID ant its name. Its intention is to *Define relative importance of criteria*. The source product part is composed of criteria organized into a set. The target product part includes also a weight class. The process part describes the main steps to follow for defining weights, namely: scale criteria according to their importance, attribute values from 1 to 100 to each criterion, and calculate relative importance. Finally, it shows how to create the corresponding class if necessary.

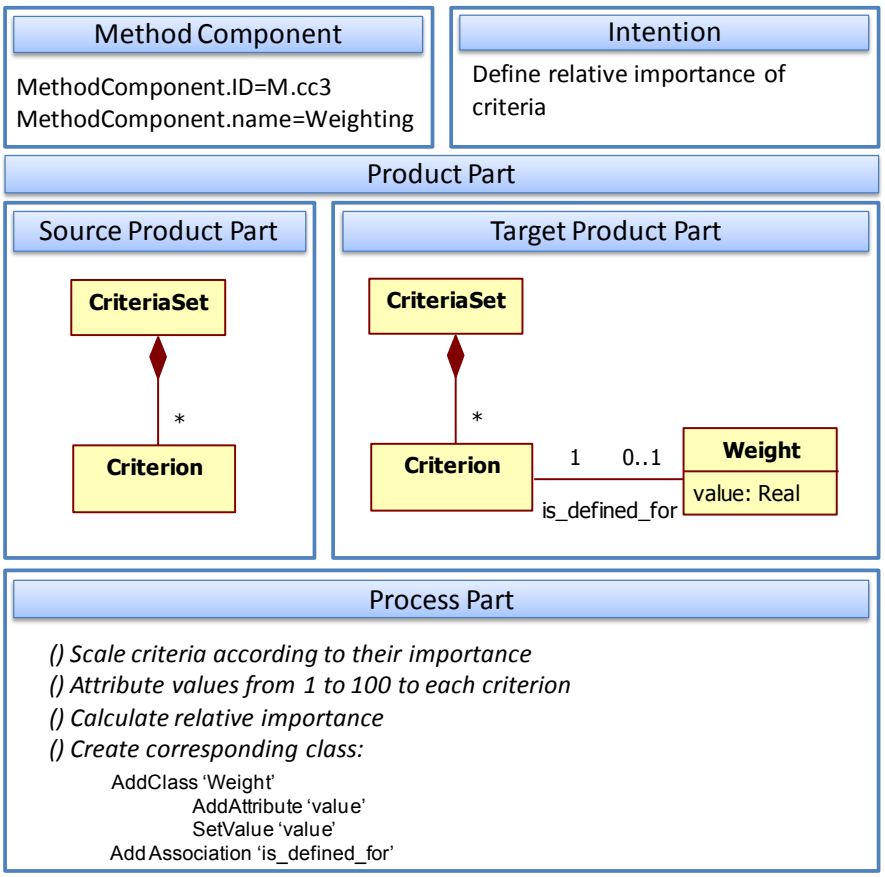

Figure 2. Decision-making Method Component Weighting.

2.3. Method Components Context

Based on the study of different SME approaches dealing with method components, we have identified five main approaches dealing with context in the method engineering field.

Reuse frame. The reuse frame (Mirbel 2008) is a framework representing different factors which affect IS development projects. These factors are called *criteria*. Reuse frame allows specifying a context of method fragments reuse, searching method fragments and comparing between them in order to find an alternative fragment to a used one. The reuse frame model includes a reuse situation (which is a set of criteria classified into three dimensions: *organizational*, *technique* and *human*) and reuse *intention*.

Interface. In (Ralyté & Rolland 2001b) the method fragment context is defined by its interface which includes a situation and an intention. The situation represents the conditions in which the method fragment can be applied in terms of required inputs product(s). The intention is a goal that the method fragment helps to achieve. Therefore, the interface model includes two elements: the *situation* and the *intention*. These two first approaches have been unified in (Mirbel & Ralyté 2006).

Method service context. The method service context (Guzélian & Cauvet 2007) aims at describing the situation in project development for which the method service is suitable and defining the purpose of the service. Its model includes *domain* characteristics (project nature, project domain) and *human* (actor), *process* and *product* ontologies.

Contingency factors. Situations (the context) are described by a set of characteristics called contingency factors (Van Slooten & Hodes 1996) or project factors (Harmsen 1997; Harmsen, Brinkkemper & Oei 1994). These factors are used to define the project situation by assigning values to them. In (Van Slooten & Hodes 1996), four categories are given: *domain characteristics* (describing the content of the system), *external factors* (laws and norms), *technical factors* (related to the development platform) and *human factors* (representing the development expertise of people).

Development situation. (Karlsson & Agerfalk 2004) defines the development situation as an abstraction of one or more existing/future software development projects with common characteristics. This situation is used to characterize the specific projects and to select configuration packages (method fragments). The development situation model includes a characteristics set.

Based on the review of these five approaches, we have identified height characteristics (context elements) which allow us to compare existing context approaches (See Table 2). This comparison highlights that there is no approach which consider all possible characteristics. Moreover, the analysis of these context approaches shows that they do not suggest a way to specify context characteristics. For instance, the context of the DM method component illustrated at Fig. 2 must be defined in order to state in what kind of situation this component is useful. However, the existing literature does not provide means for defining its context.

Table 2. Comparative analysis of approaches dealing with context in ME field: context elements.

Approach	Characteristics							
	Goal/ Intention	Organiza-tional	Technical	Human	Domain	External	Process	Product
Reuse Frame	X	X	X	X				
Interface	X							X
Method service context				X	X		X	X
Contingency factors			X	X	X	X		
Development situation	Not specified							

3. CONTEXTUALIZATION OF METHOD COMPONENTS

3.1. Our proposal

In SME, all approaches are situational, which means they take into account the specific project situation (or *Context*). However, the definition or description of this context is often just superficially addressed.

Our proposal uses the context expressiveness to describe the situation in which a component may be applied. It is then based on the *semantic* type of context previously presented. Moreover, our view of a component includes an intention oriented approach which allows representing the *cognitive* aspect of the context.

The preceding comparative analysis of context approaches shows that they address several aspects of context. However, they do not cover all of them and do not help in the context characteristics specification. Our goal is to enhance the definition of the context of IS development method for the further selection of components from a repository according to a given situation. In the following we present our vision of context and a process to define the context for a given method.

3.2. Enhanced definition of method context

We propose to consider the context granularity at two levels: the method and method component ones (See Fig. 3. for the proposal overview). Each method is available in a given context. As a method is composed of some components, each of them can be also described by specifying its context. Therefore, the method context is an aggregation of contexts associated to its components.

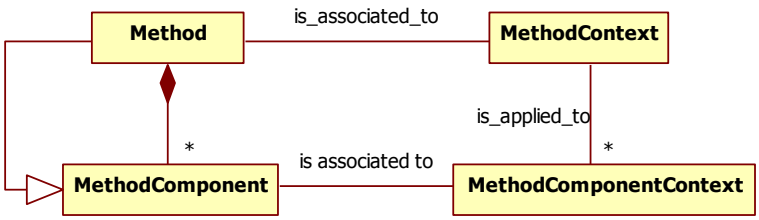

Figure 3. Proposal Overview.

In our proposal, we describe the context as a set of characteristics. These characteristics describe situations of a method application. The detailed context model is presented at Fig. 4.

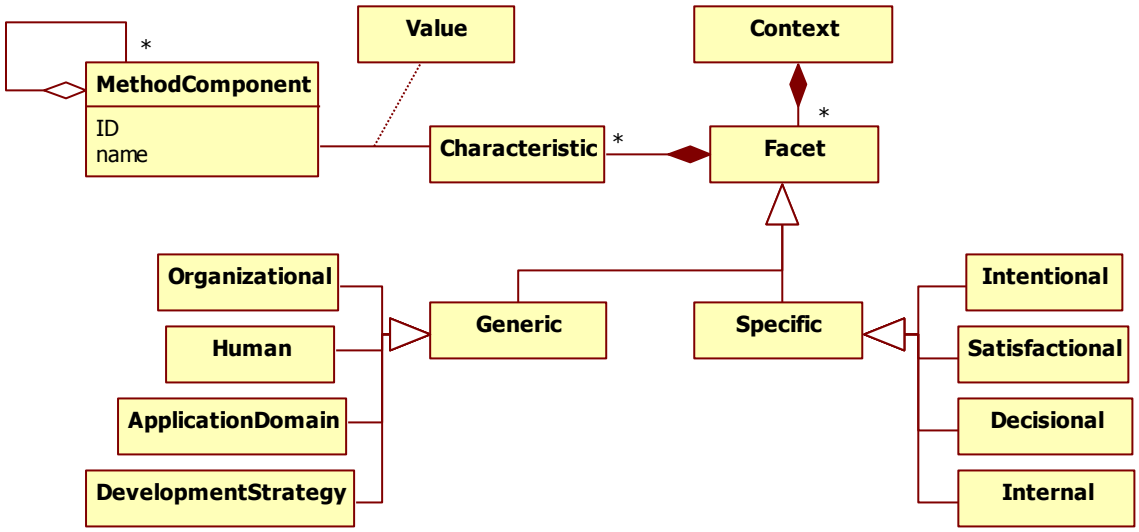

Figure 4. Context Model.

The central element of this model is characteristic. A set of characteristics constitute the context. *Context characteristics* indicate specific conditions to use the component. Characteristics are organized into *facets* for better representation and comprehension. We distinguish two types of characteristics (and consequently two types of facets): *generic* and *specific*. The first ones are common for most IS engineering projects; the latter ones vary from one project to another. To distinguish between them is important because of their different identification approaches. The context characteristics set is defined for a method component. Therefore, each method component is described by the valuations of these characteristics (*value*).

In the following, we describe different context characteristics by facets.

Generic characteristics. In order to establish the typology of generic characteristics we have used IS development project characteristics (Kornyshova, Deneckère & Salinesi 2007). In this work, a project characteristics typology is proposed in order to guide method components retrieval and to prioritize the selected components.

The suggested typology of context characteristics covers essential aspects of IS engineering projects. Based on (Mirbel & Ralyté 2006), (Van slooten & Hodes 1996) and (Kornyshova, Deneckère & Salinesi 2007), it includes four facets: organizational, human, application domain, and development strategy.

The *organizational* facet (Table 3) highlights organizational aspects of IS project development. For instance, the *Management Commitment* characteristic represents the management team involvement in the project. Possible values for this characteristic are Low, Normal and High (i.e. a High value means a high involvement and so on).

Table 3. Organizational Facet Characteristics.

Characterisitc	Type	Value domain
Management commitment degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Importance degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Impact degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Time pressure degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Shortage of resources degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Level of innovation degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Size	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Cost	Quantative	REAL
	Qualitative	ENUM:{low, normal, high}
Nature of limited resources	Qualitative	ENUM:{financial, human, temporal, informational }
Innovation nature	Qualitative	ENUM:{business innovation, technology innovation}
Duration	Quantative	REAL

The *human* facet (Table 4) describes the qualities of persons involved in IS project development. For example, the *User involvement* characteristic represents the kind of participation of the users in the project. Its values may be real or virtual.

Table 4. Human Facet Characteristics.

Characterisitc	Type	Value domain
Resistance degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Conflict degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Expertise degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Clarity degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Stability degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Expert role	Qualitative	ENUM:{tester, developer, designer, analyst}
User involvement	Qualitative	ENUM:{real, virtual}
Stakeholder number	Quantative	NUMBER

The *application domain* facet (Table 5) includes indicators characterizing the domain of IS project. For instance, the *Application type* characteristic deals with the different kinds of projects according to the organization structure and can have the following values: intra-organization application, inter-organization application, organization-customer application.

Table 5. Application Domain Facet Characteristics.

Characterisitc	Type	Value domain
Formality degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Relationships degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Dependency degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Complexity degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Repetitiveness degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Variability degree	Quantative	3-grade scale
	Qualitative	ENUM:{low, normal, high}
Application type	Qualitative	ENUM:{intra-organization, inter-organization, organization-customer }
Application technology	Qualitative	ENUM:{application to develop includes a database, application to develop is distributed, application to develop includes a GUI}
Dividing project	Qualitative	ENUM:{one single system, establishing system-oriented subprojects, establishing process-oriented subprojects, establishing hybrid subprojects}
Variable artefacts	Qualitative	ENUM:{organisational, human, application domain, and development strategy}

The *development strategy* facet (Table 6) gathers indicators about different characteristics of development strategy. For instance, the *Source system* characteristic represents the origin of the reused elements that may be code, functional domain or interface.

Table 6. Development Strategy Facet Characteristics.

Characterisitc	Type	Value domain
Source system	Qualitative	ENUM:{code reuse, functional domain reuse, interface reuse}
Project organization	Qualitative	ENUM:{standard, adapted}
Development strategy	Qualitative	ENUM:{outsourcing, iterative, prototyping, phase-wise, tile-wise}
Realization strategy	Qualitative	ENUM:{at once, incremental, concurrent, overlapping}
Delivery strategy	Qualitative	ENUM:{at once, incremental, evolutionary}
Tracing project	Qualitative	ENUM:{weak, strong}
Goal number	Quantative	NUMBER
	Qualitative	ENUM:{one goal, multi-goals}

Specific characteristics. Their identification is based on the method description. The method engineer defines them by analyzing different aspects which are organized into four facets: intentional, satisfaction, decisional and internal, like in (Harmsen 1997).

The *intentional* facet concerns the method intentions. The *satisfaction* facet indicates the satisfaction degree that the engineer has about the method application results. The *decisional* facet arises from a decision-making process in the method. The *internal* facet concerns the known criteria associated with the specific project management. For the specific map characteristics see Table 7.

Table 7. Specific Map Characteristics.

Characterisitc	Type	Value domain
Goal satisfaction degree	Quantative	3-grade scale.
	Qualitative	ENUM: {low, normal, high}
Goal achievement degree	Quantative	3-grade scale
	Qualitative	ENUM: {low, normal, high}
Section satisfaction degree	Quantative	3-grade scale
	Qualitative	ENUM: {low, normal, high}
Section completeness degree	Quantative	3-grade scale
	Qualitative	ENUM: {low, normal, high}

Table 8 shows the correspondence between the proposed typology and the existing context elements (analyzed in the previous section). We can make some remarks to compare them:

- Our typology covers all existing elements.
- We propose to identify more precisely process and product characteristics using our approach instead of using product and process as context characteristics directly.
- We add decisional characteristics which are not presented in the existing typologies.

Table 8. Correspondence between the proposed typology and existing context elements.

Proposed Typology	Context Elements (cf. Table 2)
Organizational	Organizational
Human	Human
Application domain	Domain
Development strategy	Technical
Intentional	Goal/ Intention
Satisfaction	External, Process, Product
Decisional	Process, Product
Internal	Technical, Process, Product

Our typology indicates the main characteristics that can be defined in function of a given situation. It can be completed if new characteristics arise. Fig. 5 illustrates the obtained characteristics typology as an ontology, like in (Gu, Wang, Pung & Zhang, 2004).

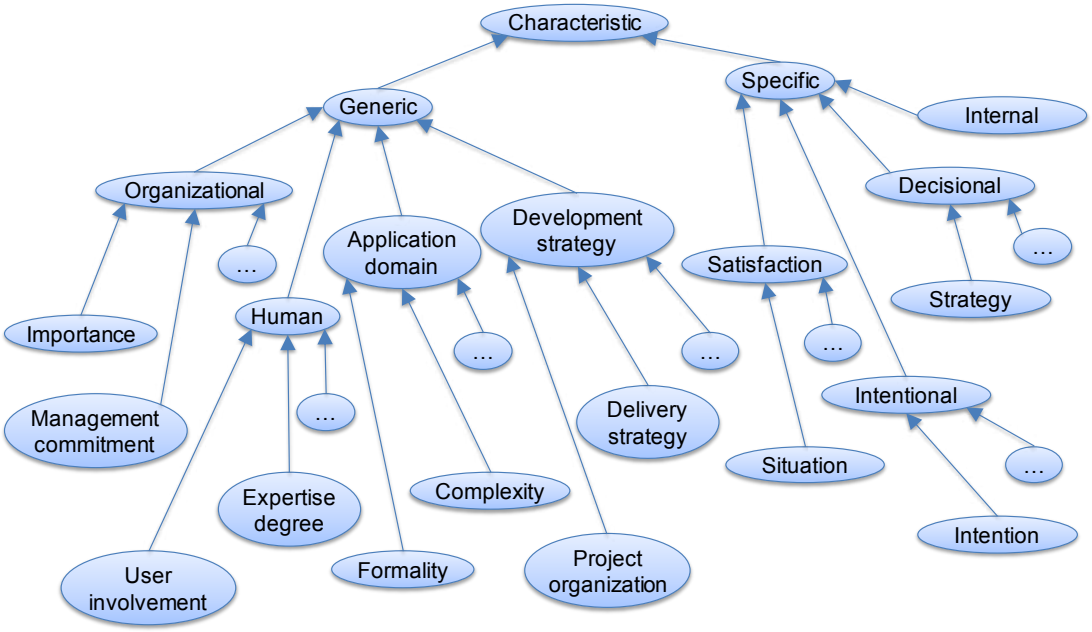

Figure 5. Characteristics ontology.

3.3. Contextualization methodology

In order to define the context for a given method and its components, we propose an approach based on the contextualization process modeled with the MAP formalism.

3.3.1. Map Formalism

A MAP illustrates a given process of IS engineering. The *MAP* model (Rolland, Prakash & Benjamen 1999) is a representation of process models expressed in intentional terms. It allows specifying process models in a flexible way by focusing on the process intentions, and on the various ways to achieve each of these intentions.

The meta-model of MAP (Rolland & Prakash 2001) is represented as a UML class diagram (See Fig. 6).

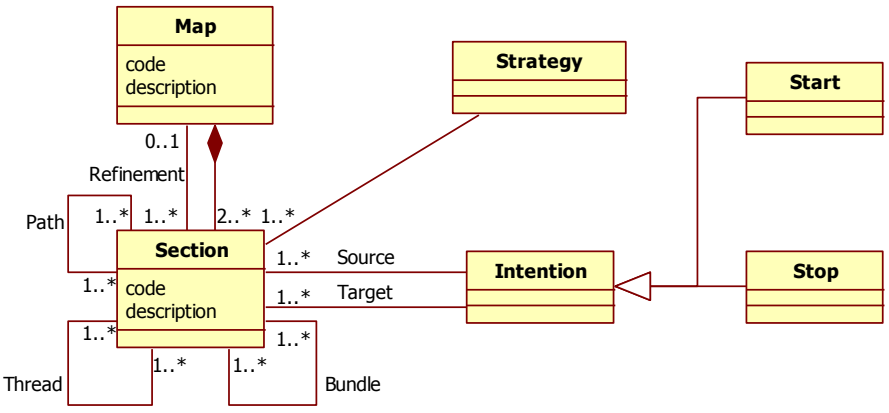

Figure 6. Map Meta-model.

A *Map* is a combination of at least two sections. Each Map has a code and a description which allow identifying it. The Map code depends on the Map position according to the refinement levels. The Map description gives a main purpose of the Map, or in other words, its main intention.

The Map model enables to represent non-deterministic sequences of activities. It is expressed through the combination of different intentions and strategies used for achieving these intentions. Actually, an engineer has several intentions or goals that he(she) wants to achieve. Furthermore, there are several ways of achieving these intentions. The Map model allows considering intentions and strategies in ISE processes in order to perform them in a flexible manner.

An *Intention* is a goal that can be achieved by the performance of an activity. Each map has two special intentions, *Start* and *Stop*, to begin and to end the map respectively (Rolland & Prakash 2001). (Prat 1997) suggest a model to describe the intention in details. This model has already been applied to the method engineering field in order to represent the intention of the method chunks (Ralyte 2001). Following this model, the intention is expressed in natural language and is composed of a verb and at least one parameter. Each parameter has a particular role with regard to the verb. An example is the *Identify DM Requirements* intention. The detailed description of the intention elements could be found in (Prat 1997) (Ralyte 2001).

A *Strategy* is an approach, a manner to achieve an intention (Rolland & Prakash 2001). Each strategy relates two intentions. The strategy concept allows, firstly, separating the goal and the manner to achieve this goal and, secondly, expressing alternative approaches for the goals achievement. An example is the *By problem exploring* strategy.

A *Section* is the main element of the Map model. It represents a combination of two intentions and a strategy relating these intentions. In other words, a section encapsulates knowledge about an activity in a triplet <Source intention; Strategy; Target intention>, in other terms, knowledge corresponding to a particular process step to achieve an intention (the target intention) from a specific situation (the source intention) following a particular technique (the strategy). A section is characterized by a code and a description. The Sections code depends on its position on the Map. The Section description embodies the triplet <Source intention; Strategy; Target intention>. An example can be mentioned: *<Start; By problem exploring; Identify DM Requirements>*.

A section of a map can be refined by another map. This is shown through the refinement relationship between the section and the map. Refinement is an abstraction mechanism by which a complex assembly of sections at level *i+1* is viewed as a unique section at level *i.* This relationship introduces levels in the process representation as each map may be represented as a hierarchy of maps.

Sections in a map are related to each other by three kinds of relationships namely thread, path and bundle.

- A *thread relationship* shows the possibility for a target intention to be achieved in several ways from the same source intention. Each of these ways is expressed as a section in the map.
- A *path relationship* establishes a precedence relationship between sections. For a section to succeed another, its source intention must be the target intention of the preceding one.
- A *bundle relationship* shows the possibility for several sections having the same source and target intentions to be mutually exclusive.

A Map is graphically presented as a directed diagram, where *intentions* are nodes and *strategies* are edges, and each *section* corresponds to two nodes related to each other by an edge (See Fig. 7). The directed nature of this diagram shows the precedence links between intentions.

An edge enters a node if its associated strategy can be used to achieve the target intention (the given node).

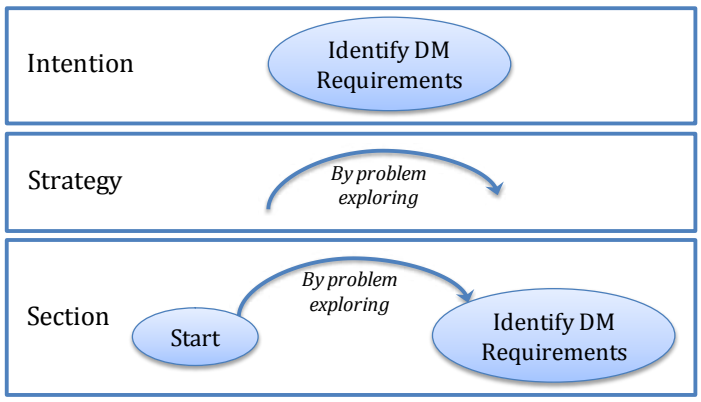

Figure 7. Map Model Graphical Representation.

An example of a map is given at Fig. 8.

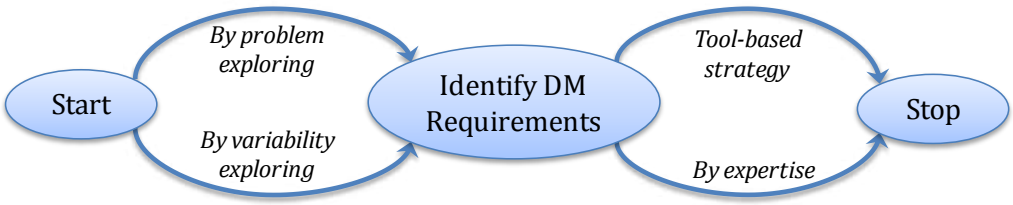

Figure 8. Map Example.

This map contains three intentions, namely: *Start*, *Identify DM Requirement*, and *Stop*. There are four strategies: *By problem exploring*, *By variability exploring*, *Tool-based strategy*, and *By expertise* which correspond to the four sections:

- *<Start; By problem exploring; Identify DM Requirements>*;
- *<Start; By variability exploring; Identify DM Requirements>*;
- *<Identify DM Requirements; Tool-based strategy; Stop>*;
- *<Identify DM Requirements; By expertise; Stop>*.

Each map is completed by a set of guidelines that help engineers in navigating through the map. There are three types of guidelines: simple, tactical and strategic. A *simple* guideline may give informal content advice on how to proceed in handling the situation in a narrative form. A *tactical* guideline is a complex guideline, which uses a tree structure to link its sub-guidelines. A *strategic* guideline is a complex guideline which shows that a section of a map can be refined by another map. This relationship implies that each map may be represented as a hierarchy of maps.

The MAP model defines the process through the combination of observable situations in which a certain number of specific intentions can be achieved. The work to be made is described in the process as depending on both situation and intention. In other words, it depends on the context in which a method engineer must act at a given point in time. By modelling *intentions* and the ways (*strategies*) to reach them, the process has the ability to represent the *cognitive*

context as defined by Bunt. Moreover, by relating method service (Rolland 2008) (or method component (Ralyté, Deneckère & Rolland 2003)) to a section, Rolland extends the context expressiveness of the MAP to the *semantic* context of Bunt. This approach allows identifying several context aspects. More precisely, this model includes a set of guidelines which help an engineer navigate through the process model. The navigation is carried out by *arguments* that allow the engineer to choose the adapted variant within the process model. These arguments express the context of a given process model.

3.3.2. Contextualization Map

The Map model is used in our approach for modeling the contextualization process (See Fig. 9). This process includes two possible ways to define the context: top-down or bottom-up. By the top-down approach, the engineer defines the method context and then instantiate it for each method component. By the bottom-up approach, the engineer specifies the contexts of all method components and assemblies them into the method context.

Both method and method component contexts can be defined following two strategies: *By deduction* and *By generation*. It depends on the characteristic type. The generic characteristics are *deduced* from the generic context typology and the specific ones are *generated* from method description. These strategies could be applied as many times as possible characteristics exist.

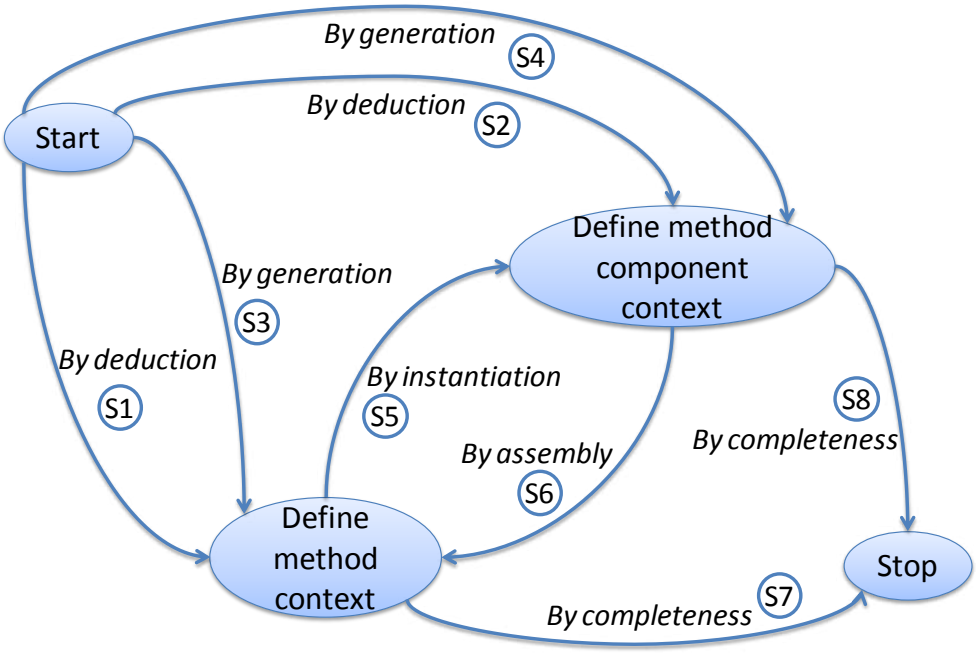

Figure 9. Contextualization Map.

This MAP has two main intentions: *Define method context* and *Define method component context*. The achievement of these intentions implies the definition of the context characteristics set for method or for method components respectively. The definition of method components contexts includes also the attribution of values to the defined characteristics.

The contextualization Map includes eight sections, as shown in Table 9.

Table 9. Contextualization Map Sections.

Section	<Source intention, Strategy, Target intention >
S_1	<Start, By deduction, Define method context>
S_2	<Start, By deduction, Define method component context>
S_3	<Start, By generation, Define method context>
S_4	<Start, By generation, Define method component context>
S_5	<Define method context, By instantiation, Define method component context>
S_6	<Define method component context, By assembly, Define method context>
S_7	<Define method context, By completeness, Stop>
S_8	<Define method component context, By completeness, Stop>

All these sections are explained below. Operators are defined for each section in order to indicate how to proceed for carrying out its execution.

<Start, By deduction, Define method context>. The generic characteristics deduction is based on the context typology. This section gives a selection of characteristics carried out by the IS method engineer. The result of this strategy is a sub-set of generic characteristics available for a given project. The corresponding operator is:

Select Context Characteristic ()

<Start, By deduction, Define method component context>. This section includes the selection of characteristics form generic typology like the previous one and includes furthermore the attribution of values to these characteristics. The result of this strategy is a sub-set of generic characteristics available for a given project with corresponding values. Two following operators are applied consecutively:

Select Context Characteristic ()
Attribute a Value to Context Characteristic ()

<Start, By generation, Define method context>. The specific characteristics generation is based on the method description. The method engineer defines them by analyzing different aspects which are organized into four facets: intentional, satisfaction, decisional and internal. This section includes four operators. Each of the following operators is applied depending on the corresponding characteristic's facet:

Analyze Method Goal ()
Measure Method Satisfaction ()
Analyze Method Argumentation ()
Measure Method Characteristics ()

<Start, By generation, Define method component context>. The definition of specific characteristics for method components context is the same as for method context (the previous section) but also requires the attribution of characteristics values. This section uses the same four operators and adds another one that deals with the attribution of values to the characteristics. This last one is applied after each of the first four operators for defining concrete values of the identified specific characteristics.

Analyze Method Goal () [for intentional facet]

Measure Method Satisfaction () [for satisfaction facet]
Analyze Method Argumentation () [for decisional facet]
Measure Method Characteristics () [for internal facet]
Attribute a Value to Context Characteristic () [for all facets]

<Define method context, By instantiation, Define method component context>. The context characteristics instantiation is common for both characteristics types and is applied in the top-down approach. This section allows defining a sub-set of generic and specific method characteristics with an associated value for each method component separately. Several functions may be applied (sum, maximum, minimum, average, weighted sum, and so on) for attributing values to the method context. This section contains two operators applied consecutively:
Retain Context Characteristic ()
Attribute a Value to Context Characteristic ()

<Define method component context, By assembly, Define method context>. In the case of the bottom-up approach, the strategy *By assembly* follows the definition of the method component context *By deduction* or *By generation*. The method engineer groups method components characteristics together. As a result, the method context includes all characteristics of its components contexts. The application of this strategy allows also defining characteristics' values for the method context. This section is carried out by the following operator:
Group Characteristics ()
Attribute a Value to Context Characteristic ()

<Define method context, By completeness, Stop> and **<Define method component context, By completeness, Stop>.** These sections are the same in both top-down and bottom-up approaches and include verification of completeness and coherence of the described context. The associated operator is:
Verify Context Completeness ()

All these operators are resumed in the Table 10.

Table 10. Operators' Description.

Operators	Description
Select Context Characteristic ()	Helps to select each of the pertinent characteristics of the context.
Attribute a Value to Context Characteristic ()	For each characteristic selected, a value corresponding to the project context has to be defined.
Analyze Method Goal ()	Helps to define the characteristics of the intentional facet which concerns the method intentions (the method goals).
Measure Method Satisfaction ()	Helps to measure the satisfaction degree on the results obtained by the engineer and concerns the satisfaction facet.
Analyze Method Argumentation ()	The decisional facet needs this operator in order to describe a decision-making situation with the definition of the arguments to take into account in the DM process.
Measure Method	This operator is used to give values to the characteristics associated with

Characteristics ()	the specific project management.
Retain Context Characteristic ()	Helps to define a subset of characteristics (generic or specific characteristics) and to give them values adapted to the project.
Group Characteristics ()	This operator allows to group all the characteristics together in the same set which corresponds to the context.
Verify Context Completeness ()	Helps to study the completeness and the coherency of the context set of characteristics.

4. APPLICATION: THREE CASES STUDIES

In this section, we illustrate our proposal by applying the contextualization methodology to three cases: scenario conceptualization, project portfolio management and decision-making.

We use the Map model for representing methods and for organizing method components into methods in our examples. The key concept of a Map is the notion of Section. When dealing with methods modeled by maps, each method component is represented by a map section. Thus, each Map section is linked to a particular method component, as shown in Fig. 10.

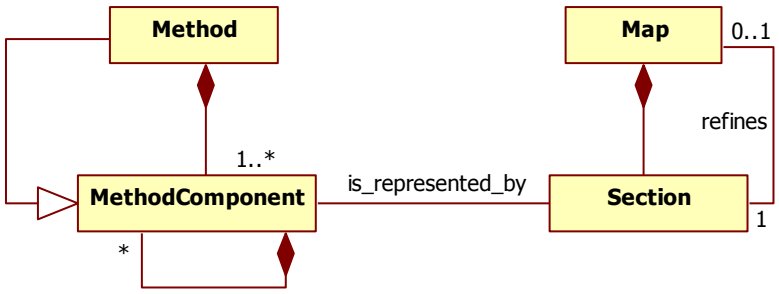

Figure 10. Section and Method Component Correspondence.

Each case study describes a particular map organizing method components. We then show how the engineer may use the contextualization map in order to create the context of the method/ components.

4.1. Case Study #1: Scenario Conceptualization

4.1.1. Case Study Description

The first example is based on the map defined in the Crews-L'écritoire approach (Ralyté, Rolland, Plihon &Ralyté 1999). This map is given at Fig. 11.

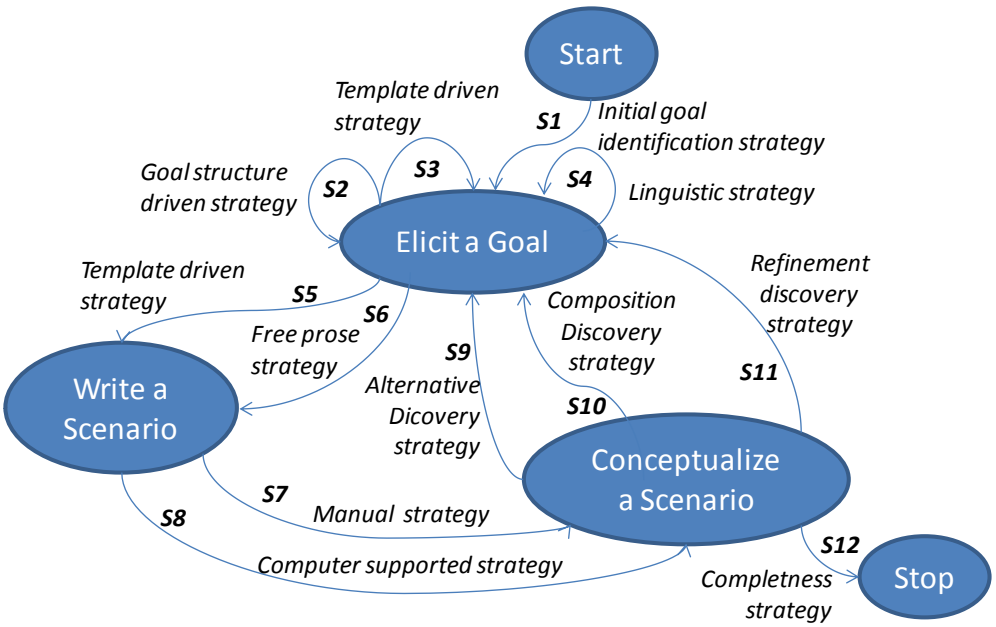

Figure11. Crews-L'écritoire Approach Map (CL Map).

This map was created to support the elicitation of functional system requirements in a goal-driven manner and to conceptualize them using textual devices such as scenarios or use cases. It provides guidelines to discover functional system requirements expressed as goals and to conceptualize these requirements as scenarios describing how the system satisfies the achievement of these goals. This map contains three main intentions, namely 'Elicit a Goal', 'Write a Scenario' and 'Conceptualize a Scenario'. There are twelve separate sections which allow the engineer to navigate through the map. Each of these sections is expressed as a specific component which is saved into a repository.

4.1.2. Contextualization Process

The engineer executes the contextualization map. He decided to select the top-down approach of the contextualization process. This specific way to navigate through the map will guide the engineer first through the definition of the method characteristics and then through the definition of the components ones. Fig. 12 shows the path used in the navigation through the map in order to define the method components contexts.

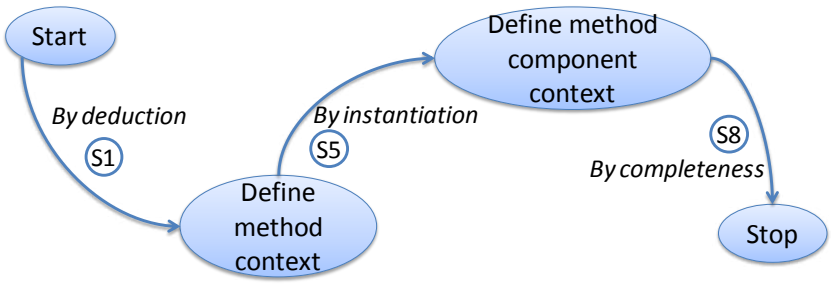

Figure 12. Path used in the Contextualization Map for the CL Map Case Study.

This selected path contains three sections of the Contextualization Map and may be resumed on the three following steps:

1. Definition of the method Context (S_1),
2. Definition of the method components contexts (S_5), and
3. Verification of the context completeness (S_8).

First step (*S_1: Definition of the method context).* The engineer had studied the set of possible generic characteristics to specify the context of its method. He has applied the operator *Select Context Characteristic ()* in order to define a sub-set of generic context characteristics according to the given project. In this example, he has selected three indicators: *Expertise degree*, *Formality degree* and *Duration* (See Table 11).

Table 11. Generic facet values.

Characteristic	Value domain
Human facet	
Expertise degree	3-grade scale
Application domain facet	
Formality degree	3-grade scale
Organisational facet	
Duration	REAL

Second step (S_5: *Definition of the method components context*). The method context defined previously is instantiated in this step for each component (each section of the CL map). It means that a value has to be affected to each characteristic. The following operators are applied to each method characteristic: *Retain Context Characteristic ()* and *Attribute a Value to Context Characteristic ()*. Table 12 shows the values of these criteria applied to each section of the CL map.

Table 12. CL map indicators

Section		*Expertise degree*	*Formality degree*	*Duration*
S1	<Start; Initial goal identification strategy; Elicit a goal>	1	1	10 mn
S2	<Elicit a goal; Goal structure driven strategy; elicit a goal>	1	2	15 mn
S3	<Elicit a goal; Template driven strategy; elicit a goal>	1	3	15 mn
S4	<Elicit a goal; Linguistic strategy; elicit a goal>	1	1	15 mn
S5	<Elicit a goal; Template strategy; write a scenario>	2	3	10 mn
S6	<Elicit a goal; Free prose strategy; write a scenario>	1	1	15 mn
S7	<Write a scenario; Manual strategy; conceptualize a scenario>	2	1	15 mn
S8	<Write a scenario; Computer supported strategy; conceptualize a scenario>	1	3	5 mn
S9	<Conceptualize a scenario; Alternative discovery strategy;	1	1	20 mn

	Elicit a Goal>			
S10	<Conceptualize a scenario; Composition discovery strategy; Elicit a Goal>	2	2	20 mn
S11	<Conceptualize a scenario; Refinement discovery strategy; Elicit a Goal>	2	2	20 mn
S12	<Conceptualize a scenario; Completeness strategy; Stop>	1	1	5 mn

Third step (S_8: *Verification of the context completeness*). The engineer has decided that the identified context characteristics are sufficient by applying the operator *Verify Context Completeness ()*.

4.2. Case study #2: Project Portfolio Management

4.2.1. Case Study Description

The second example deals with Information Technology Project Portfolio Management (IT-PPM). The Fig. 13 shows the IT-PPM intentional map which describes the ways to manage a project within a portfolio.

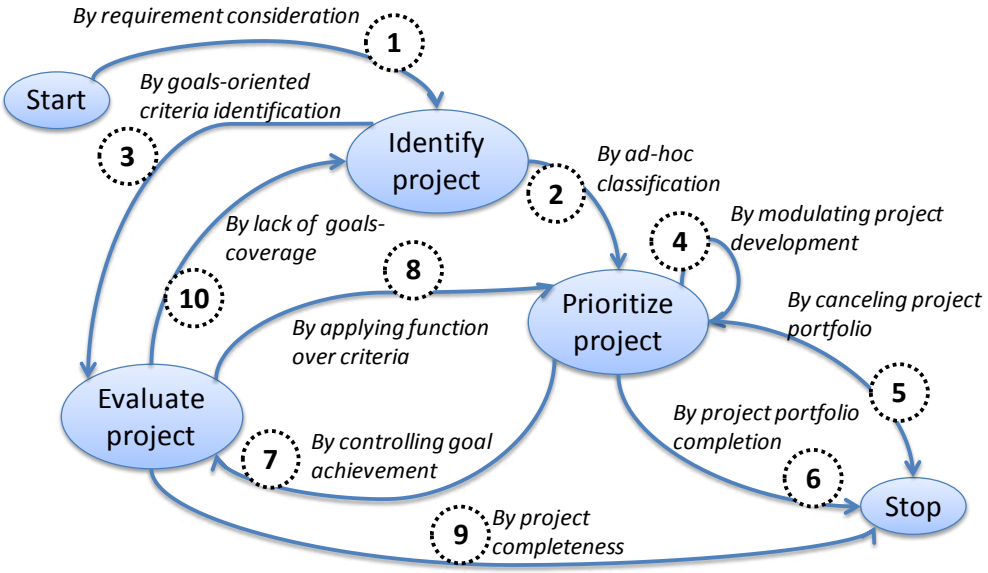

Figure 13. IT-PPM Map.

This Map is a refinement of the section < Define Risks, By Project Planning, Align IT and Business Process > of the map dealing with IT governance presented in (Claudepierre & Nurcan 2009). This map contains three main intentions: 'Identify project', 'Evaluate project', and 'Prioritize project' and ten associated sections. The related components are saved into a method base which includes their description and methodological guidelines for their application.

4.2.2. Contextualization Process

The engineer has selected the top-down approach of the contextualization process. It guides the engineer through the definition of the method characteristics before the definition of method component characteristics. Fig. 14 shows the path used in the navigation through the contextualization map in this particular case study.

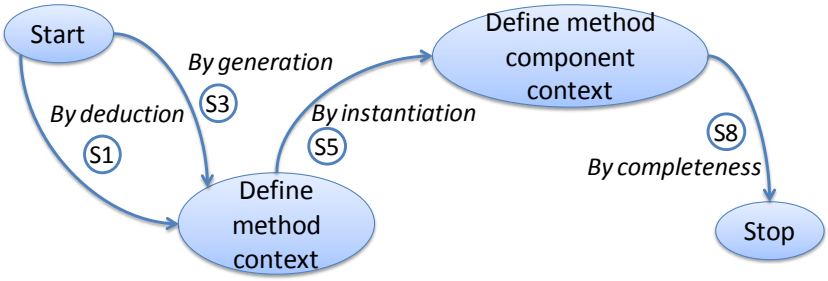

Figure 14. Path used in the Contextualization Map for the IT-PPM Case Study.

This path contains four sections of the Contextualization Map that we represent within the three following steps:

1. Definition of the method Context (S_1 and S_3),
2. Definition of the method components contexts (S_5), and
3. Verification of the process completeness (S_8).

First step (*S_1, S_3: Definition of Method Context*). This step contains the execution of two sections of the Contextualization Map: *S_1* and *S_3*.

Definition of the generic characteristics *(S_1).* The engineer uses the characteristics presented in Table 4 and Table 5 as generic characteristics to specify the context of the method. The engineer has applied the operator *Select Context Characteristic ()* in order to define a sub-set of generic context characteristics according to the given project. He has selected three generic characteristics for this example: *Expertise degree*, *Expert role* and *Application type* (Table 13).

Table 13. Generic Characteristics.

Characteristic	Value domain
Human facet	
Expertise degree	{low, normal, high}
Expert role	{tester, developer, designer, analyst}
Application domain facet	
Application type	{intra-organization application, inter-organization application, organization-customer application}

Definition of the specific characteristics *(S_3).* The engineer also uses *specific* context characteristics (cf. Table 14). This specific context is depicted by the constraints of the business environment (the design *situation*), the *intention* of the designer and the *strategy* for reaching the intention. So, the three operators were applied to identify the specific characteristics. *Analyze Method Goal ()* is used to identify the *Intention* which is related to the intentional type of specific characteristic. *Measure Method Satisfaction ()* allows defining the *Situation* in the Satisfactional

facet (as it describes the satisfaction degree of the previous intention). Finally, *Analyze Method Argumentation ()* defines the *Strategy* in the decisional facet of the specific characteristic.

Table 14. Specific Characteristics.

Characteristic	Value domain
Intentional facet	
Intention	TEXT
Satisfactional facet	
Situation	TEXT
Decisional facet	
Strategy	TEXT

Second step (S_5: Definition of Method Components Context). The method context defined at the previous step is now instantiated for each component. A value is affected to each characteristic in order to help the case process execution guidance. The following operators are applied to each method characteristic: *Retain Context Characteristic ()* and *Attribute a Value to Context Characteristic ()*. The results are presented in Table 15.

Table 15. Specific Characteristics Instantiation.

Section		*Expertise degree*	*Expert role*	*Appli. Type*	*Situation*	*Intention*	*Strategy*
S1	<Start; By goals-oriented criteria identification; Identify project>	Normal	Analyst designer	Intra-Org.	Problem statement	Identify project	By requirement consideration
S2	<Identify project; By ad-hoc classification; Prioritize project>	Low	Analyst designer	Intra-Org.	Project identified	Prioritize project	By ad-hoc classification
S3	<Identify project; By goals-oriented criteria identification; Evaluate project>	High	Analyst designer	Intra-Org.	Project identified	Evaluate project	By goals-oriented criteria identification
S4	<Prioritize project; By modulating project development; Prioritize project>	High	Designer	Intra-Org.	Project prioritized	Prioritize project	By modulating project development
S5	<Prioritize project; By cancelling project portfolio; Stop>	Low	Designer	Intra-Org.	Project prioritized	Stop	By canceling project portfolio
S6	<Prioritize project; By project portfolio completion; Stop>	Low	Designer	Intra-Org.	Project prioritized	Stop	By project portfolio completion
S7	<Prioritize project; By controlling goal achievement; Evaluate project>	Low	Designer	Intra-Org.	Project prioritized	Evaluate project	By controling goal achievement
S8	<Evaluate project; By applying function over criteria; Prioritize project>	Normal	Analyst	Intra-Org.	Project evaluated	Prioritize project	By applying function over criteria
S9	<Evaluate project; By project completeness; Stop>	Low	Designer	Intra-Org.	Project evaluated	Stop	By project completeness
S10	<Evaluate project; By lack of goals-coverage; Identify project>	Normal	Analyst	Intra-Org.	Project evaluated	Identify project	By lack of goal coverage

Third step (S_8: Verification of the process completeness). The engineer has decided that the identified context characteristics are sufficient to allow a satisfying guidance through the portfolio project management by the operator *Verify Context Completeness ()* application.

4.3. Case Study #3: Decision-Making

4.3.1. Case Study Description

The third example allows defining context for the Decision-making (DM) generic process. The map model of the DM generic process (Kornyshova 2010) is given at Fig. 15.

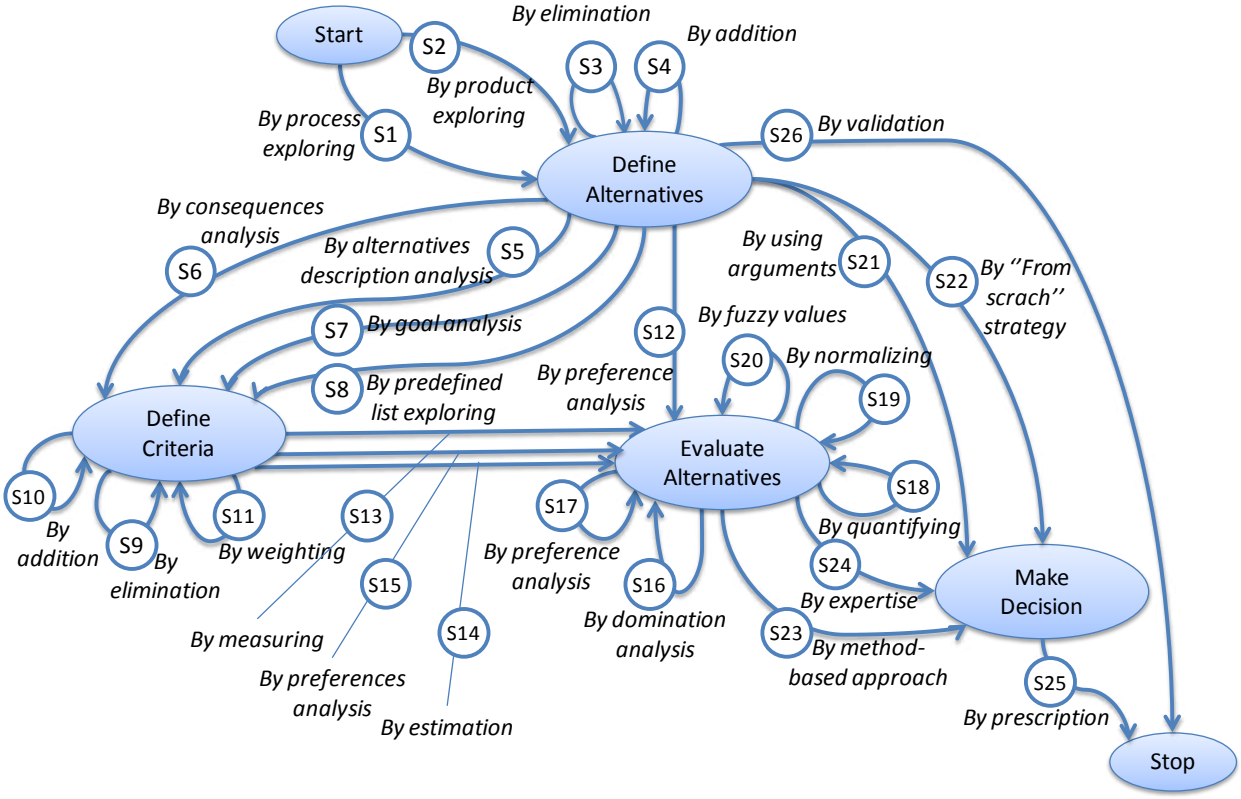

Figure 15. DM Generic Process Map (DM Map).

The DM Method Family describes the generic DM process including the main activities used for DM. It can be used each time an IS engineer meets a DM situation. The DM map is a collection of DM method components organized into a generic process (a kind of multi-method) for their easier usage in practice. DM components represent detailed guidelines for DM activities associated to the specific context of their use. The DM map contains four main intentions: 'Define Alternatives', 'Define Criteria', 'Evaluate Alternatives', and 'Make Decision' and twenty six sections, or DM method components.

4.3.2. Contextualization Process

The engineer executes the contextualization map and selects the bottom-up approach of the contextualization process as he is able to qualify each DM component. In this case, the engineer specifies the contexts of all method components and assemblies them into the method context. The used path in the Contextualization Map is shown at Fig. 16.

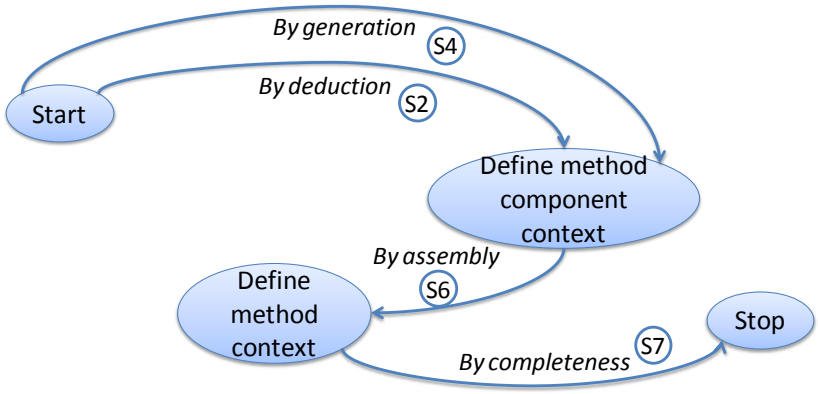

Figure 16. Path used in the Contextualization Map for the DM Map Case Study.

This selected path contains four sections of the Contextualization Map and may be resumed on the three following steps:

1. Definition of the method components contexts (S_2 and S_4),
2. Definition of the method Context (S_6), and
3. Verification of the context completeness (S_7).

First step (*S_2, S_4: Definition of Method Component Context*). This step contains the execution of two sections of the Contextualization Map: *S_2* and *S_4*.

Definition of the generic characteristics *(S_2)*. The engineer uses the characteristics presented in Table 4 and Table 5 as generic characteristics to specify the context of the DM components. He has applied the operator *Select Context Characteristic ()* in order to define a sub-set of generic context characteristics according to the given project. The engineer has selected two generic characteristics: *Expertise degree*, and *Complexity degree* (See Table 16).

Table 16. Generic characteristics.

Characteristic	Value domain
Application domain facet	
Complexity degree	3-grade scale
Human facet	
Expertise degree	3-grade scale

Then, the engineer attributes values to each DM method components using the *Attribute a Value to Context Characteristic ()* operator (See Table 18).

The engineer has attributed values to twenty three DM components. The other three components (N/E – not evaluated values in Table 18) cannot be evaluated according to these characteristics as they depend significantly on the given situation.

Definition of the specific characteristics *(S_4).* The engineer also uses *specific* context characteristics (cf. Table 17) leading him to qualify method components. An operator was applied to identify the specific characteristics: *Analyze Method Goal ()* for identifying the *Intention*.

Table 17. Specific Characteristic.

Characteristic	Value domain
Intentional facet	
Intention	TEXT

The engineer also attributes an intention to each DM method components using the *Attribute a Value to Context Characteristic ()* operator (See Table 18).

Table 18. Specific Characteristics Instantiation.

Section		*Complexity degree*	*Expertise degree*	*Intention*
S1	<Start; By process exploring; Define alternatives>	2	2	Define alternative list
S2	<Start; By product exploring; Define alternatives>	2	2	Define alternative list
S3	<Define alternatives; By elimination; Define alternatives>	1	1	Refine alternative list
S4	<Define alternatives; By addition; Define alternatives>	1	1	Refine alternative list
S5	<Define alternatives; By alternatives description analysis; Define criteria>	2	1	Define criteria list
S6	<Define alternatives; By consequences analysis; Define criteria>	2	1	Define criteria list
S7	<Define alternatives; By goal analysis; Define criteria>	1	1	Define criteria list
S8	<Define alternatives; By predefined list exploring; Define criteria>	1	1	Define criteria list
S9	<Define criteria; By elimination; Define criteria>	1	2	Refine criteria list
S10	<Define criteria; By addition; Define criteria>	1	2	Refine criteria list
S11	<Define criteria; By weighting; Define criteria>	2	1	Define relative importance of criteria
S12	<Define alternatives; By preferences analysis; Evaluate alternatives>	2	1	Evaluate alternatives
S13	<Define criteria; By measuring; Evaluate alternatives>	2	1	Evaluate alternatives
S14	<Define criteria; By estimation; Evaluate alternatives>	2	1	Evaluate alternatives
S15	<Define Criteria; By preferences analysis; Evaluate alternatives>	2	1	Evaluate alternatives
S16	<Evaluate alternatives; By domination analysis; Evaluate alternatives>	2	2	Discard dominated alternatives
S17	<Evaluate alternatives; By preferences analysis; Evaluate alternatives>	2	2	Refine alternative evaluations

S18	<Evaluate alternatives; By quantifying; Evaluate alternatives>	1	2	Quantify alternative values
S19	<Evaluate alternatives; By normalizing; Evaluate alternatives>	1	2	Normalize alternative values
S20	<Evaluate alternatives; By fuzzy values; Evaluate alternatives>	3	3	Define fuzzy values of alternatives
S21	<Define alternatives; By using arguments; Make decision>	N/E	N/E	Make decision
S22	<Define alternatives; By "From scratch" strategy; Make decision>	N/E	N/E	Make decision
S23	<Evaluate alternatives; By method-based approach; Make decision>	3	3	Make decision
S24	<Evaluate alternatives; By expertise; Make decision>	N/E	N/E	Make decision
S25	<Make decision; By prescription; Stop>	1	2	Prescribe decision
S26	<Define alternatives; By validation; Stop>	1	3	Validate decision

Second step (S_6: *Definition of the method context).* In order to define the context characteristics of the DM method, the engineer uses the *Group Characteristics ()* operator. For this, he takes all characteristics, specified for all components. Then he applies the *Attribute a Value to Context Characteristic ()* operator for evaluating the method context characteristics.

In order to attribute values to the method context (See Table 19), he chooses the maximal value for the characteristics Complexity degree and Expertise degree, and for intention, he takes the main intention which is the Make Decision intention (This intention is on the top of the taxonomy of the DM intentions.).

Table 19. Method Context Values.

Characteristic	Value
Complexity degree	3
Expertise degree	3
Intention	Make Decision

Third step (S_8: *Verification of the context completeness*). The engineer has decided that the identified context characteristics are sufficient by applying the operator *Verify Context Completeness ()*.

4.4. Lessons Learned

The three case studies have shown that the contextualization approach can be used in various areas of information system engineering.

Closely related to the method at hand, it requires the strong degree of the engineer commitment into methodological processes. The engineers' intervention is highly required at all steps of the contextualization process. In addition, it allows taking into account the specificity of each studied method, that is to say the context-awareness. The last one is one of the most important issues in the current science of IS engineering. In this manner, these three examples show how the contextualization approach can contribute to resolve this issue.

The main usages of the contextualization results are the following: a more simple navigation through the map (in the case of Scenario conceptualization); a better 'context-aware' selection of method components (in the case of PPM); an easier customization of the generic process (in the

case of DM); and, finally, means for identifying situations in which a given component is useful (three case studies). For instance, the *Weighting* DM method component (See Fig. 2) is useful in the situation characterized by the level 2 of complexity (normal), requiring the level 1 of expertise (low), and when the goal is to define the relative importance of criteria.

However, these case studies have made obvious that it is more easily to use the generic characteristics of context then to try to find some specific characteristics. It means that the operators for identifying specific characteristics must be enhanced.

5. RELATED WORKS

The current work was motivated by a need to formalize an approach for specifying context of methods and method components. It is related to the following fields of information system engineering: situational method engineering (SME), decision-making in information system engineering, and process variability.

SME approaches. Several works has been done to define the concept of method component in order to obtain flexible methods. The different kind of method components present in the literature are the method fragment (Brinkkemper 1996), the method chunk (Ralyté, Deneckère & Rolland 2003), the method component (Wistrand & Karlsson 2004), the OPF fragment (Henderson-Sellers 2002) and the method service (Guzélian & Cauvet 2007). Some details on these method components are given in section 2.2. In this field, the paper contributes to the methodology of identifying and evaluating method context characteristics.

Decision-making methods in ISE. With regard to IS engineering, the issue of DM has already been explored with respect to requirements engineering [NgoTheAl2005], to method engineering [Aydin2006], and, more generally, to systems engineering [Ruhe2003]. Ruhe emphasizes the importance of DM in SE along the whole life cycle [Ruhe2003]. However, DM in IS engineering has several lacks: (i) decisions are not formalized in terms of alternatives and criteria, their consequences are not analyzed, decisions are not transparent, (ii) at intuitive and ad hoc decisions overshadow method-based ones, (iii) and there is no tool which covers a complete DM process even if DM tools exist. To overcome these drawbacks, some studies are made, for instance a generic DM process is proposed (Kornyshova 2010) and an ontology of the DM concepts is elaborated (Kornyshova & Deneckère 2010).

Process variability and the MAP process model. Variability has proved to be a central concept in different engineering domains to develop solutions that can be easily adapted to different organizational settings and different sets of customers at a low price. The MAP formalism has a high level of variability as it is expressed in an intentional manner through goals and strategies. As a high level of variability means a high number of variation points, a process customization is then required to offer a better guidance. In a parallel way to the Product lines concept which has appeared within the management of variability and customization of products, a new concept has arise to represent the processes that may be customized to a given project: the *Process lines* (Deneckère & Kornyshova 2010a). In (Deneckère & Kornyshova 2010b), Maps are considered as Process lines and a typology of characteristics is used to configure the line in order to obtain a process adapted to the project at hand.

6. DISCUSSION AND CONCLUDING REMARKS

The situational method engineering field aims at considering methods as a set of method components. Different approaches have been defined to consider this concept of method component (method fragment, method component, method services, method chunk, and OPF

fragment). Each of these approaches mainly focus on the definition of what is a method component and how to assemble them in order to create a new method adapted to the project at hand. All of these approaches hint the fact that the notion of context has to be used to enhance the method component retrieving as they use several context related notions (interface, contingency factors, development situation, and so on). However, the process of how to identify and evaluate the method context is not suggested and our proposal is (i) to give a strong definition of a method component context and (ii) to offer a contextualization process which will help engineers to define the method components context with ease.

Strong definition of a method component context. We have studied the literature in order to define the criteria that may be used to characterize the situation in which method components may be used. This leads us to define a typology of characteristics which we have structured in different facets (each considering a special view of a project). We then related these characteristics to the method component concepts.

Contextualization process. We have identified two possible ways to use these characteristics for defining context (the top-down and the bottom-up approaches) in order to propose a contextualization process that may be adapted to several situations. We modeled this process with the MAP formalism in order to keep a high level of flexibility in the process utilization.

This proposal can be applied in different IS engineering situations such as the selection of a component for enhancing the existing IS engineering method (for instance, extension-based approaches) or a selection of several components for constructing a new one (for instance, assembly-based approaches).

We have applied the proposed model on three case studies as follows.

- **Scenario Conceptualization.** This case is based on a well know process used on the project *'Crews L'Ecritoire'*. The case study use the contextualization process to help the engineer to navigate through the process and select the right components following its degree of expertise, the duration of each performed component and its formality degree (which are generic characteristics of the typology).
- **IT Project Portfolio Management.** This case study contributes to the study of the relatively unexplored domain of IT governance from the SME point of view. The engineer selects more characteristics than in the first case study as he chooses also specific characteristics (characteristics to apply on a specific process model, in this case the MAP process model).
- **Decision making.** The contribution of this case study is twofold: the validation of the contextualization methodology and the application of the SME principles to a field issue from the operational research. Firstly, the DM case study has shown how to describe the context of DM components using three characteristics (complexity degree, expertise degree, and intention) and to identify the method context from the context of its components. Secondly, this case has demonstrated that the SME approach (identification of method components and their contextualization) is successfully applied to the DM methods for their further utilization in the IS engineering field.

Our future work aims at: (i) enhancing the approach for a more simple identification of specific context characteristics; (ii) ensuring the adaptability of methods with regards to the context specificities; and (iii) proposing a method for a formalized selection of method components following their characteristics values.

REFERENCES

Bessai K., Claudepierre B., Saidani O. & Nurcan S. (2008). Context-aware business process evaluation and redesign. In proceedings of the international workshop BPMDS'08.

Bouquet P., Ghidini Ch., Giunchiglia F. & Blanzieri E. (2003) Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics*, 35(3).

Bradley N. A. & Dunlop M. D. (2005). Toward a multidisciplinary model of context to support context-aware computing. *Human-Computer interaction*, Lawrence Erlbaum Associates.

Brinkkemper S. (1996). Method engineering: engineering of information systems development method and tools. *Information and Software Technology Journal,* 38:7.

Bunt H. (1997). Context and dialogue control. In Proceedings of CONTEXT'97.

Claudepierre B. & Nurcan S. (2009). ITGIM: An intention driven approach for analyzing the IT governance requirements. In proceedings of the international Workshop on Requirements, Intentions and Goals in Conceptual Modeling.

Coutaz J. & Rey G. (2002). Recovering foundations for a theory of contextors. In proceedings of the 4th ICCADUI, Valenciennes, France.

Dey A., Abowd G. & Salber, D. (2001). A conceptual framework and toolkit for supporting the rapid prototyping of context-aware applications, *Human-computer Interaction*, 16 2-4 (Special issue on context-aware computing), (pp. 97–166).

Deneckère R. , Iacovelli A. , Kornyshova E. & Souveyet C. (2008). From method fragments to method services. In proceedings of the Evaluation of Modeling Methods in Systems Analysis and Design conference (EMMSAD'08), Montpellier, France.

Deneckère R. & Kornyshova E. (2010a) La variabilité due à la sensibilité au contexte dans les processus téléologiques, *Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*, Marseille, France (In French).

Deneckère R. & Kornyshova E. (2010b). Process Line Configuration: an Indicator-based Guidance of the Intentional Model MAP, Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD), Hammamet, Tunisie.

Drury J. L. & Scott S. D. (2008). Awareness in unmanned aerial vehicle operations. *International C2 journal*, Geoffrey N. Hone, 2:1.

Firesmith D. & Henderson-Sellers B. (2001). The OPEN Process Framework. An Introduction, Addison-Wesley.

Gonzales-Perez C. (2007). Supporting situational method engineering with ISO/IEC 24744 and the work product tool approach. In proceedings of the International IFIP WG8.1 Conference ME 07, Springer, Geneva, Switzerland.

Gu T., Wang X.H., Pung H.K. & Zhang D.Q. (2004), An Ontology-based Context Model in Intelligent Environments. In proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, (pp 270-275).

Guzélian G. & Cauvet C. (2007). SO2M : Towards a service-oriented approach for method engineering. In proceedings of the international conference IKE'07, Las Vegas, Nevada, USA.

Harmsen F. (1997). *Situational method engineering*. Moret Ernst & Young.

Harmsen A.F., Brinkkemper J.N. & Oei J.L.H. (1994). Situational method engineering for information systems project approaches. In proceedings of the international IFIP WG8. 1 Conference in CRIS series : "Methods and associated Tools for the Information Systems Life Cycle" (A-55), North Holland (Pub.).

Henderson-Sellers B. (2002). Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering, 14(1-4).

Iacovelli A., Souveyet C. & Rolland C. (2008). Method as a service (MaaS). In proceedings of the International Conference on Research Challenges in Information Science (RCIS'08), Marrakech, Morocco, (pp. 371 – 380).

Karlsson F. & Agerfalk P.J. (2004). Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology 45, (pp 619-633).

Kirsch Pinheiro M., Vanrompay Y. & Berbers Y. (2008). Context-aware service selection using graph matching. In proceedings of ECOWS 2008, vol. 411.

Kornyshova E., Deneckère R. & Salinesi C. (2007). Method chunks selection by multicriteria techniques: an extension of the assembly-based approach. In proceedings of the International IFIP WG8.1 Conference ME 07, Springer, Geneva, Switzerland.

Kornyshova E. & Deneckère R. (2010). Decision-Making Ontology for Information System Engineering", *International Conference on Conceptual Modeling (ER)*, Vancouver, Canada.

Mirbel I. (2008). Contributions à la modélisation, la réutilisation et la flexibilité des systèmes d'information . HDR thesis, Nice University.

Mirbel I. & Ralyté J. (2006). Situational method engineering: combining assembly-based and roadmap-driven approaches. In proceedings of the international conference Requirements Engineering (RE'06), 11(1), (pp. 58–78).

Mirbel I. & de Rivieres V. (2002) Adapting Analysis and Design to Software Context : The jecko Approach, In 8th International Conference on Object Orirented Information Systems.

Nehan Y. R. & Deneckère R. (2007). Component-based situational methods - A framework for understanding SME. In proceedings of the International IFIP WG8.1 Conference ME'07, Springer, Geneva, Switzerland.

Prat N. (1997). *Goal formalisation and classification for requirements engineering*. Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97, Barcelona (pp. 145-156).

Ralyte J. (2001). Method chunks engineering, PhD thesis, University of Paris 1-Sorbonne.

Ralyté J., Deneckere R. & Rolland C. (2003). Towards a generic model for situational method engineering. In proceedings of the international conference CAISE'03, Springer Verlag, Velden, Austria.

Ralyté J. & Rolland C. (2001a). An assembly process model for method engineering. In proceedings of the International Conference on Advanced information Systems Engineering (CAISE'01), Interlaken, Switzerland.

Ralyté J. & Rolland C. (2001b). An approach for method reengineering. In proceedings of the 20th International Conference on Conceptual Modeling (ER'01), Yokohama, Japan, November 2001. H. Kunii, S. Jajodia, A. Solvberg (Eds.), LNCS 2224, Springer-Verlag, (pp.471-484).

Ralyté, J., Rolland, C. & Plihon, V. (1999) Method Enhancement with Scenario Based Techniques. Proceedings of the 11th International Conference on Advanced Information System Engineering (CAISE'99), Heidelberg, Germany, M. Jarke, A. Oberweis (Eds.), LNCS 1626, Springer-Verlag, (pp. 103-118).

Rey G. & Coutaz J. (2002). Le Contexteur : une abstraction logicielle pour la réalisation de systèmes interactifs sensibles au contexte. *IHM'02*, (pp. 105-112).

Rolland C. & Cauvet C. (1992). Object-Oriented Conceptual Modelling, CISMOD'92, International Conf. on Management of Data, Bangalore.

Rolland C., Plihon V. & Ralyté J. (1998). Specifying the reuse context of scenario method chunks. In proceedings of the international conference CAiSE'98, Pise, Italy.

Rolland C., Prakash N. & Benjamen A. (1999). A multi-model view of process modeling. In proceedings of the Requirements Engineering international conference (RE'99), Springer-Verlag London Ltd, 4:4.

Rolland C. & Prakash N. (2001) Matching ERP System Functionality to Customer Requirements. In Procs of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada. August 27-31.

Rolland C. (2005). L'ingénierie des méthodes : une visite guidée. *E-revue en Technologies de l'Information (e-TI)*, Invited Talk.

Rolland C. (2008). Method engineering: "Towards methods as services". In proceedings of the International Conference on Software Process (ICSE-ICSP), Springer-Verlag, Leipzig, Germany.

Rosemann M. & Recker J. (2006). Context-aware process design: exploring the extrinsic drivers for process flexibility. In proceedings of workshops and doctoral consortium in the 18th international conference on advanced information systems engineering. Luxembourg: Namur University Press., (pp 149-158).

Rosen M. A., Fiore S. M., Salas E., Letsky M. & Warner N. (2008). Tightly coupling cognition: understanding how communication and awareness drive coordination in teams. International C2 journal, 2:1.

Schilit B., Adams N. & Want R. (1994). Context-aware computing applications. In proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Santa Cruz, CA, US, 89-101.

Van Slooten K. & Hodes B. (1996), Characterising IS development projects. In proceedings of the IFIP WG8.1 Conference on Method Engineering.

Wistrand K. & Karlsson F. (2004). Method components: rationale revealed. In proceedings of the international conference CAISE'04, Springer-Verlag. Riga, Latvia.

APPENDIX

Elena Kornyshova

Elena Kornyshova is a PhD student at the University of Paris 1 Panthéon-Sorbonne (CRI - Centre de Recherche en Informatique) under the direction of Pr. Colette Rolland and Dr. Rébecca Deneckère. Her research domains are Method Engineering, Process Engineering, Enterprise Architecture and Decision-making in Information System Engineering. Her PhD research aims to propose a Method Engineering approach to improve Decision-making in Information System Engineering.

Rébecca Deneckère

Rébecca Deneckere is affiliated to the CRI (Centre de recherche en Informatique) at the university of Paris 1 Panthéon-Sorbonne. Her domain of research is the Method Engineering field, especially Situational Method Engineering. She is also working on Decision-making in Information System Engineering . Her last field of research is the processes context-awareness and the configuration of method lines.

Bruno Claudepierre

Bruno Claudepierre is a PhD student at the University of Paris 1 Panthéon-Sorbonne (CRI - Centre de Recherche en Informatique) under the direction of Pr. Colette Rolland and Dr. Selmin Nurcan. His research purposes are focused on Information Systems engineering methods and their adaptations in order to comply with the new requirements of IT Governance. He usually works with CRI staff members on connected research areas like Business Process Redesign, Method Engineering, Business/IT alignment and Information System Design.

In order to access to up to date information about authors, please scan the following codes.


Elena Kornyshova


Rébecca Deneckère


Bruno Claudepierre

For scanning abilities, you may install the following barcode scanner software on your mobile: http://www.lynkware.com/support_devices.php